COMPUTER CORPORATION OF AMERICA

An Experimental Computer Network

March 30, 1969

Prepared for
Massachusetts Institute of Technology
Lincoln Laboratory

An Experimental Computer Network

Prepared by
Computer Corporation of America

For
Massachusetts Institute of Technology
Lincoln Laboratory

Under
Purchase Order No. C-538
Prime Contract No. AF 19(628)-5167
ARPA Order No. 691

This Report Covers the Period
January 1, 1967 to March 31, 1969

Abstract

The concept of a computer network is reviewed. Within such a network, a user at any one installation has immediate access to the hardware, software, and data at all other installations. The various installations need not be program-compatible.

The implementation of a small experimental network based on these concepts is discussed. The details of the implementation and the experience gained in the operation of this experimental network are described.

Accepted for the Air Force
Franklin C. Hudson
Chief, Lincoln Laboratory Office

Table of Contents

Chapter 1
Introduction

1.1 The Concept of a Computer Network

Within a computer network, as we conceive of it, a user at any one installation has immediate access to the hardware, programs, and data at all other installations. The various installations within this network need not be program-compatible.

As an initial example of how a network may be used advantageously, consider the following. As is well known, it is much more expensive to invert a large matrix on a small computer than on a large computer. The user of a small machine, however, usually has no alternative but to bear the extra cost, particularly if his matrix inversion routine is part of a larger program running on his computer. Under the network, on the other hand, his program can, at run time, ship the matrix to be inverted to a large machine in the network, cause the large machine to run its own inversion program on the data shipped, receive the inverse from the large machine, and go on running. Note that it was data, not program, which was shipped over the network, and that the two computers in this example could have been program-incompatible.

As a second example, consider the case of a user of computer A who wants to interrogate a data-bank stored at computer B. In the ordinary situation, his options are (a) to have the data-bank copied and shipped to him (by mail, say) to be loaded on his own computers, or (b) to run his interrogation program on computer B. Neither alternative is really satis-

factory. Alternative (a) will take a long time, and the data-bank may be too large to fit on computer A. Alternative (b) is out of the question if the interrogation routine is part of some larger program running at A.

Under networking, on the other hand, the program running on A causes, at run time, an interrogation program to run at B; this program sends back the answers to A, and the main program goes on running, using the answers it has received.

The examples point up the following potential advantages of networks.

1. Increased capability. To have one's computer tied into a network increases one's access to data, programs, and hardware.

2. More efficient allocation of computer resources. Since each network participant has access to all computers in the network, and since we may assume that these computers cover a range of sizes and specialities, the network provides a way of designing each program for a computer suitable to it. Different subroutines of the same program may run on different computers, if specialization in these computers can be used to advantage. As a result, the efficiency (that is to say, the computation per unit cost) increases throughout the network. It may be taken to be one of the underlying conjectures of the networking approach that the increase in efficiency which accrues in this way more than compensates for the communication costs involved in operating the network itself. This conjecture is still to be tested.

3. Distributed data banks. Data may be kept in various

places in the network, and yet be available to all users. In principle, at least, any piece of data need be represented only once in the network, rather than once at each installation which needs it. This fact leads to economies in storage space, and perhaps of greater importance, to a decreased file-maintenance problem--since it is easier to maintain one file on a given machine than ten copies of this file on ten- potentially incompatible and geographically separate- machines.

4. Decreased duplication of effort and increased coordination. Users of any installation tend to be insular; understandably so, since the programs developed at other installations rarely do them any good. In the network, however, the remote installations become accessible and their programs useable. Hence, one may expect that more interest will be shown in work going on at these other installations, leading to improved communication and cooperation between users. The availability of, and interest in, other people's work will decrease the problem of duplication of effort.

These various advantages do not, of course, come for free. Among the disadvantages are the following:

1. Increased communication costs. In an ordinary time-sharing situation, the standard costs include the communication between users and computer. In a network, the additional costs of computer-computer communication lines must also be considered.

2. Greater complexity in software and hardware systems. The software to handle computer-to-computer communication, particulary at high data rates, represents a substantial addition to existing operating systems or time-sharing monitors. Also required are modifications to handle bookkeeping, protection, and resource allocation. Suitable hardware to interface

between computer and communication lines must be provided.

3. Increased need for documentation. It is sometimes possible to make do with marginally adequate documentation, if all users form a closely-knit group. If however, some users are across the continent, it is imperative that good documentation be provided.

1.2 Approach to Implementation

There are many different ways of approaching the problem of implementing a computer network. Our approach is oriented to the following goals and concepts.

1. A program on computer A should be able to call a program on computer B with no more trouble than another program on A itself.

2. Programs should not be shipped from one computer to another. Only data and calls to programs should be transmitted.

3. In forming a network among a set of computers, the computers and their software systems should be disturbed as little as possible. In other words, the work of forming the network should proceed as much as possible within existing systems.

4. Time-shared systems, by their nature, have the capability of handling multiple communication lines, and are therefore the easiest to tie into a network. Hence, network-formation should start with time-shared systems.

1.3 The Experimental Network

The notion of a computer network in the sense discussed above has been under consideration for some time. A preliminary study was undertaken in 1965-1966 [1]*. On the basis of recommendations in this preliminary study, plans were made for the implementation of a small experimental network. These plans were discussed in a paper [2] presented at the 1966 Fall Joint Computer Conference. Since that time the plans have, to an extent, been implemented; the remainder of this report is devoted to a description of this work, to a discussion of certain techniques used and others that were not used, and to the experience gained to date.

---

* Numbers in square brackets refer to entries in the bibliography.

Chapter 2
Hardware

2.1 Overall Network Configuration

Three computer installations were involved:

1. The TX-2 at MIT Lincoln Laboratory in Lexington, Massachusetts.
2. The AN/FSQ-32 (Q-32 for short) at System Development Corporation in Santa Monica, California.
3. A Digital Equipment Corporation (DEC Model 338) at the Pentagon, in Washington, D.C.

The TX-2 and the Q-32 are very large machines which provide general-purpose time-sharing services. The DEC 338 is a small, non-time-shared, display-oriented computer.

The overall configuration of the network is given in Fig. 1.

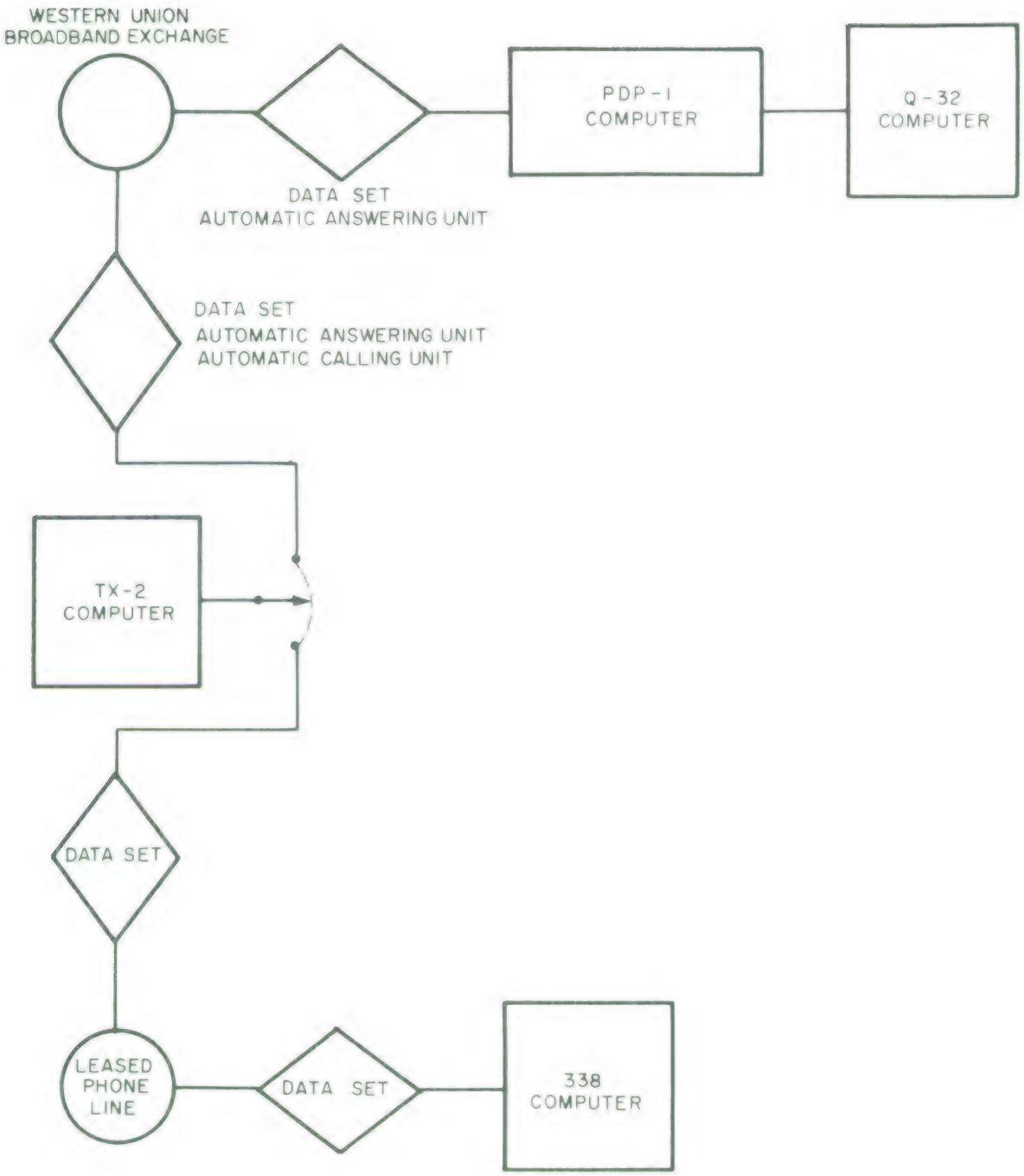

Figure 1. Experimental Network Configuration

For communication between the TX-2 and the Q-32, the Western Union broadband exchange network is used. This is a dial-up network, similar to the telephone network (see Section 2.5). For communication between the TX-2 and the 338, a private (leased) telephone connection is used (see Section 2.5).

2.2 The TX-2 Installation

The TX-2 is a large research computer built by (and located at) the MIT Lincoln Laboratory, Lexington, Massachusetts. The TX-2 has an exceptionally flexible input and output system and complement of I-O devices. The machine can accommodate a large number of in-out "sequences" similar to data channels [11].

For telecommunication purposes, a special I-O channel facility has recently been implemented. This facility has the capability for accommodating up to about 30 low-speed devices. The channel can handle a total of 10,000 characters per second.

The interface between this channel and external communication equipment is provided by autonomous data terminals. These data terminals include double-ranked parallel data buffers, registers for holding operating modes, and provide the facility for character assembly and disassembly.

The data terminals currently installed were used for interfacing with:

1. Western Union Broadband Exchange service.
2. A leased telephone line to Washington, D.C.
3. Data-phone service for use with remote teletypes.

The first two of these are used for the experimental network.

2.3 The Q-32 Installation

System Development Corporation in Santa Monica, California operates a large time-sharing system based on two interconnected computers: the AN/FSQ-32, a large military IBM computer, and a PDP-1 computer manufactured by Digital Equipment Corporation.

The PDP-1, which serves as a communications controller for the Q-32, was originally modified by the addition of a 256 channel sequence break system and a DEC Model 630, which contains the character buffers for remote user lines and the associated logic.

The lines are handled strictly on a half-duplex basis in that (a) the transmitted characters are shifted into the receive buffer for echo checking, and (b) there is only one interrupt routine for each channel (each pair of buffers).

The time-sharing system can service some 50 lines simultaneously. Most of these lines are used for in-house teletypes. Several of them connect to the nationwide TWX and Data-Phone service.

For purposes of the network, some additions were made to the PDP-1 hardware. One teletype line-interface was modified to run at 1200 bits per second and to interface with Western Union Broadband Exchange service. In addition, an interface was provided for an automatic calling unit and automatic answering unit.

2.4 The 338 Installation

The third element of the network consists of a Digital Equipment Corporation Model 338. The 338 is essentially a DEC PDP-8 computer coupled to a programmed, buffered display device.

The display shares the memory of the PDP-8 and operates on display information which has been stored in memory. The display information contains control commands, which are executed by the display logic, as well as the display data itself.

For purposes of the network, a modified DEC teletype interface (LTO-8) capable of handling 1200 bit-per-second asynchronous communication was added to the 338.

2.5 Communication Media and Auxiliary Equipment

The TX-2 and Q-32 communicate via the Western Union Broadband Exchange service. This service provides for 4-wire dial-up connections* at various bandwidths: the one use is 4KC (analogue) bandwidth. The tariff for this service is as follows:

First minute	$0.75**
Each additional 0.1 minute	$0.075

The following discount policy applies: If, in any month, the usage charges exceed $500, then the amount of such excess is reduced by 40%.

The fixed monthly charges at each end (including local loop and modem) come to approximately $100.

The following Western Union equipment is used in conjunction with the Western Union service:

Equipment	Model Number
Modems	Western Union 2121B
Automatic calling units	Western Union 12405A
Automatic answering units	Western Union 11774A

These modems provide for 1200 bit-per-second asynchronous transmission in full-duplex mode.

---

* The Telephone company does not provide 4-wire dial-up service.

** These tariffs were in effect at time of writing.

The TX-2 and the 338 communicate via a leased voice-grade telephone line (schedule 4C). This line provides a permanent (non-dial-up) 4-wire connection. The standard tariff for the line is approximately $670. At each end, the local channel, channel terminal, and modem rent for approximately $200 per month.

Chapter 3
Message Format and Protocol

3.1 Communication Conventions

Before two computers--or two people--can communicate, certain conventions regarding the communication process must be agreed upon. These communication conventions can be roughly divided into the categories of protocol and message format.

Message format deals with the basic elements by which communicated information is expressed and the manner in which these elements are combined to form more extended communications. The problems of message format are quite technical and have received considerable attention.

Protocol deals with the overall procedure for exchanging messages. Included are such questions as how messages are to be acknowledged, how the receiver is to ask for a retransmission if a message is garbled, etc. The problems of protocol are well illustrated in the following passage [3]:

> "Now, where were we? Read me back the last line."
> " 'Read me back the last line,' " read back the corporal, who could take shorthand.
> "Not my last line, stupid!" the colonel shouted. "Somebody else's."
> " 'Read me back the last line,' " read back the corporal.
> "That's my last line again!" shrieked the colonel, turning purple with anger.
> "Oh, no, sir" corrected the corporal. "That's my last line. I read it to you just a moment ago. Don't you remember sir? It was only a moment ago."

3.2 ASCII and Related Proposals

The ASCII Proposal

It is customary to consider a message as decomposable into characters, each of which represents a small number of bits. This idea is a carry-over from the days in which data-communication usually meant teletype-communication, and the purpose of a message was to cause the terminal device to print out a sequence of graphic symbols. This carry-over, as will become apparent, is unfortunate.

Teletype devices use 8-bit character codes; seven bits carry information and one is ignored. The proposed American Standard Code for Information Interchange (ASCII) is also an eight-bit character format. Seven of the eight bits represent a character code and one is used for a parity bit [4], [5].

In the 7-bit ASCII code, there are, of course, 128 characters. Of these, 32 are "control characters" which do not represent graphic symbols.

For example, the character 0000111 is called BEL. If transmitted to a teletype, BEL rings a bell.

Another character, 0010110, is called SYN, and is supposed to help the transmitter and receiver synchronize their transmissions.

Regarding the eighth bit, the parity bit, it is recommended [5] that it be odd parity for synchronous data communication and even parity for asynchronous communication.

The preceding defines the proposed standard character code.

The ASCII proposal for message format is as follows [6].

First transmit an SOH (start of heading) character, then a sequence of characters called a heading, then an STX (start of text), then the text (information to be transmitted), then an ETX (end of text).

Thus, a message looks like this:

```
S           S       E
O Heading   T  Text  T
H           X       X
```

The heading of the message contains routing information and is defined to be "a sequence of characters which constitute the auxiliary information necessary to the communication of a text" [6]. The text is the information to be communicated. It can be any sequence of characters except "communication control characters".

If we wish to break up text into "blocks", we do this as follows:

```
S           S       E S         E
O Heading   T  Text  T T  Text   T
H           X       B X         X
```

It will be recalled that 32 out of the 128 ASCII characters are called "control characters". Of these 32, ten are designated to be communication controls and are used to control the communication process. The ten consist, first, of SOH, STX, and ETX, which frame the message itself. In addition, there are seven intralink control characters which "pass

between the two ends of a link to aid in administering the transmission of a message". In our terminology, these define the protocol. They are:

EOT (End of transmission)
ENQ (Enquiry)
ACK (Acknowledge)
NAK (Negative acknowledge)
ETB (End of transmission block)
SYN (Synchronous idle)
DLE (Data link escape).

The use to which these intralink control characters are to be put is not made precise.

The message format given above allows the transmission of all "text" which does not contain the ten forbidden characters. Since, however, we may wish to transmit completely arbitrary binary information, corresponding, for example, to a pictorial display, provision is made in ASCII for a transparent mode.

The ASCII proposal for handling "transparent text", i.e., text in which any character is permissible, is as follows [7]:

1. The characters in the message to be transmitted are sent as 7-bit characters, with the eighth bit set to the usual parity.

2. Control characters, in the case of transparent mode transmissions, are distinguished by having their parity bit set the wrong way. Thus, no 8-bit byte occurring in the message itself can look like a transparent-mode control character.

3. However, the above scheme is prone to error, says ASCII, since a transparent-mode ASCII character could be generated from a regular character through a single-bit error. "The method chosen to avoid dilution of single-error detection capability is that of representing each of the additional (transparent-mode) controls by a sequence of two characters from the 'wrong' parity subset."

Three of the transparent-mode control characters are defined to be

DLE	STX
DLE	ETX
DLE	ETB

where the underline is used to represent wrong parity.

In transparent-mode, then, a message looks like this

S		D	S		D E
O	Heading	L	T	Text	L T
H		E	X		E X

or, in the case of blocks, like this

S		D	S		D	E	D	S		D	E
O	Heading	L	T	Text	L	T	L	T	Text	L	T
H		E	X		E	B	E	X		E	X

In the case of transparent-mode, the "text" consists of a sequence of 8-bit bytes, one of the bits being reserved for parity, the other seven being arbitrary.

Finally, it should be noted that ASCII allows for the eventuality that longitudinal error detection may also be used.

A "checksum" may be calculated for an entire message and appended to the end, or else may follow each block.

The Eisenbies Proposal

J. L. Eisenbies proposes [12] to handle the problem of binary transmission (transparent mode) differently. It is proposed to send 8-bit bytes (without parity bit). To provide for control characters, one of the regular ASCII characters, namely DLE (with normal parity), is set aside. The DLE is then placed in front of the usual STX, ETB, and ETX character, just as in the ASCII transparent mode (except that these characters now have the conventional parity).

The problem now comes up regarding characters in the binary message that just happen to look like DLE. In the Eisenbies technique, such 8-bit bytes are detected on transmission and doubled up. On reception, one of every pair of DLE's within the binary message is deleted.

Bhushan and Stotz Proposal

An alternative proposal, made by Bhushan and Stotz [8], is to transmit 6-bit binary bytes for binary transmission. The binary transmission is preceded and followed by special control characters. In a control character, every six-bit byte is preceded by a 1, guaranteeing that the 7-bit character so formed cannot be interpreted as an ASCII control character (since all of these start with 0). This technique, as Bhushan and Stotz point out, "is ASCII-conforming, easy to implement, and keeps aside the standard control character set for error control and recovery protocol. Further, it is independent of code-sensitive equipment, and does not require insertion and deletions of DLE's in the transmitted and received texts."

3.3 Comments on ASCII and Related Proposals

It does not appear that the ASCII approach is particularly pertinent to computer-to-computer communication. The fact that 0000111 means BEL and is supposed to ring a bell on certain terminals is not pertinent, nor is the fact that ETX or DLE may be interpreted in certain ways by certain equipment. The only clearly pertinent criterion, it would seem to us, is the efficient transmission of binary information; none of the ASCII-related proposals meet that criterion.

First, as regards ASCII itself, only seven out of every eight bits transmitted carry information; thus 12.5% of the bandwidth is wasted. Proponents of ASCII argue that the eighth bit, being used for parity-checking, serves a useful purpose. But does it?

A parity bit can, at best, serve to detect single-bit errors. A parity bit is therefore useful if the probability of single-bit errors is very much higher than the probability of other errors. This is not the case in high-speed transmission.

Thus, in 1200 bit-per-second data transmission, there is a 23% chance that, given a bad bit, the next bit will also be bad [9].

If we were to assume that a character could be changed by error into any other character with equal probability, then a parity bit would allow only 50% of all errors to be detected, which would be virtually useless.

In addition to losing 12.5% efficiency, the ASCII scheme constrains us to dealing in 7-bit bytes at the transmitting and

receiving computers. Since a sizable fraction of the computing community deals in 8-bit bytes, the ASCII proposal imposes quite a considerable inconvenience (packing eight-bit bytes into seven bit ones on transmission and unpacking on reception) on this fraction of the community.

The Eisenbies proposal allows for the transmission of 8-bit bytes and is therefore not subject to the above criticism. However, the gain of efficiency is counterbalanced by a considerable increase in complexity. Every character of the binary message itself must be checked by the transmitter to determine if it is DLE, and if so, an extra DLE must be inserted. This processing does not come for free; hence, we obtain full efficiency (8 out of 8 bits carry information), but only at the price of extra hardware or software at the receiver and transmitter.

The Bhushan and Stotz proposal gets around some of the complexities of the Eisenbies proposal. However, their scheme transmits only six information bits out of every eight, and therefore wastes a full 25% of the bandwidth. This appears to us to be quite unacceptable.

The conclusions to be drawn from all this depends, no doubt, on the point of view of the person drawing the conclusions. From our point of view, the conclusion seems clear enough: ASCII is quite unsuited to the case of computer-to-computer communication and should be abandoned for computer networks.

There is, in fact, a certain element of unreality in the ASCII-related proposals. Does Eisenbies seriously put forward the notion, as a standard to be followed from here on out, that all bit streams be checked on transmission to determine whether

they happen to contain the character 0010000 (in order then to insert special bits which are later removed on reception)? Should we seriously entertain, as Bhushan and Stotz suggest, the notion of wasting 25% of our bandwidth merely in order to be ASCII-conforming? These questions seem to us to prove, by reductio ad absurdum, how ill-suited the ASCII concepts are for the case of computer-to-computer communication.

3.4 Message Format Used in Experimental Network

In the experimental network, eight data bits are sent per character. All information transmitted is sent in the form of messages. No character parity is used; a checksum covering the entire message serves for error detection.

A message consists of a header character, the body of the message, an end-of-message character, and a checksum.

In transmitting eight-bit binary information, the first character in the message body is a count, equal to the total number of characters in the body (including the count character itself). In the present implementation, the count must be less than 119.

The header character for binary messages is an octal 221 or octal 232 according as the message is addressed to the monitor program at the receiver or to a user program. The end-of-message character is an octal 203. (These three characters correspond to ASCII DC1, SS, and ETX, respectively, with the high order bit set to 1.) The checksum is the 8-bit ring sum of the header character and characters in the body.

Alphanumeric characters may, of course, be transmitted using the 8-bit binary (transparent) mode. Thus ASCII characters, parity bit and all, may be transmitted. In addition, a special alphanumeric mode is also provided, in which all characters of the message body are specifically interpreted by the receiver as alphanumeric characters. The parity bit is not used in this case.

In the alphanumeric mode, the high order bit of each character

in the body is set to 0; this serves to distinguish these characters from control characters, which have the high order bit set to 1. There is no count character, and up to 119 characters may be transmitted.

The header character for alphanumeric messages is an octal 201 or 202 according as the message is addressed to the monitor program at the receiver or to a user program. The end-of-message character is an octal 203. (These three characters correspond to ASCII SOH, STX, and ETX, respectively, with the high order bit set to 1.) The same checksum, covering the header, body, and end-of-message, is used as in the binary case.

The message format described allows for eight-bit binary transmission. It is straightforward, and does not require the scanning of the binary message to insert and remove special characters.

3.5 Message Protocol Used in Experimental Network

The process is as follows: A message is sent out by the transmitter. The transmitter then waits for a reply before sending out the next message. There are four possible replies:

1. Message received OK (octal 206). The same character is also used to say "go ahead", after a wait (see below).

2. Message received in error (octal 225).

3. Message received OK, but buffers are full--wait before sending next message (octal 234).

If no reply is received in one second, a query (octal 230) is sent out to determine if the message or the reply was lost. The effect of the query is to request the receiver to repeat the last reply sent to it.

Similarly if no "go ahead" (octal 206) is received within 30 seconds of a wait, a query (octal 230) is sent out.

The process is flow-charted in Fig. 2.

Three additional characters are reserved for special functions. The first is sync (octal 226), which is used for establishing synchronization when synchronous modems are used on the TX-2/338 line. When asynchronous modems are used, the sync characters are ignored. The other two characters--help (octal 220) and panic (octal 233)--are both treated as a break at SDC or a help request at TX-2.

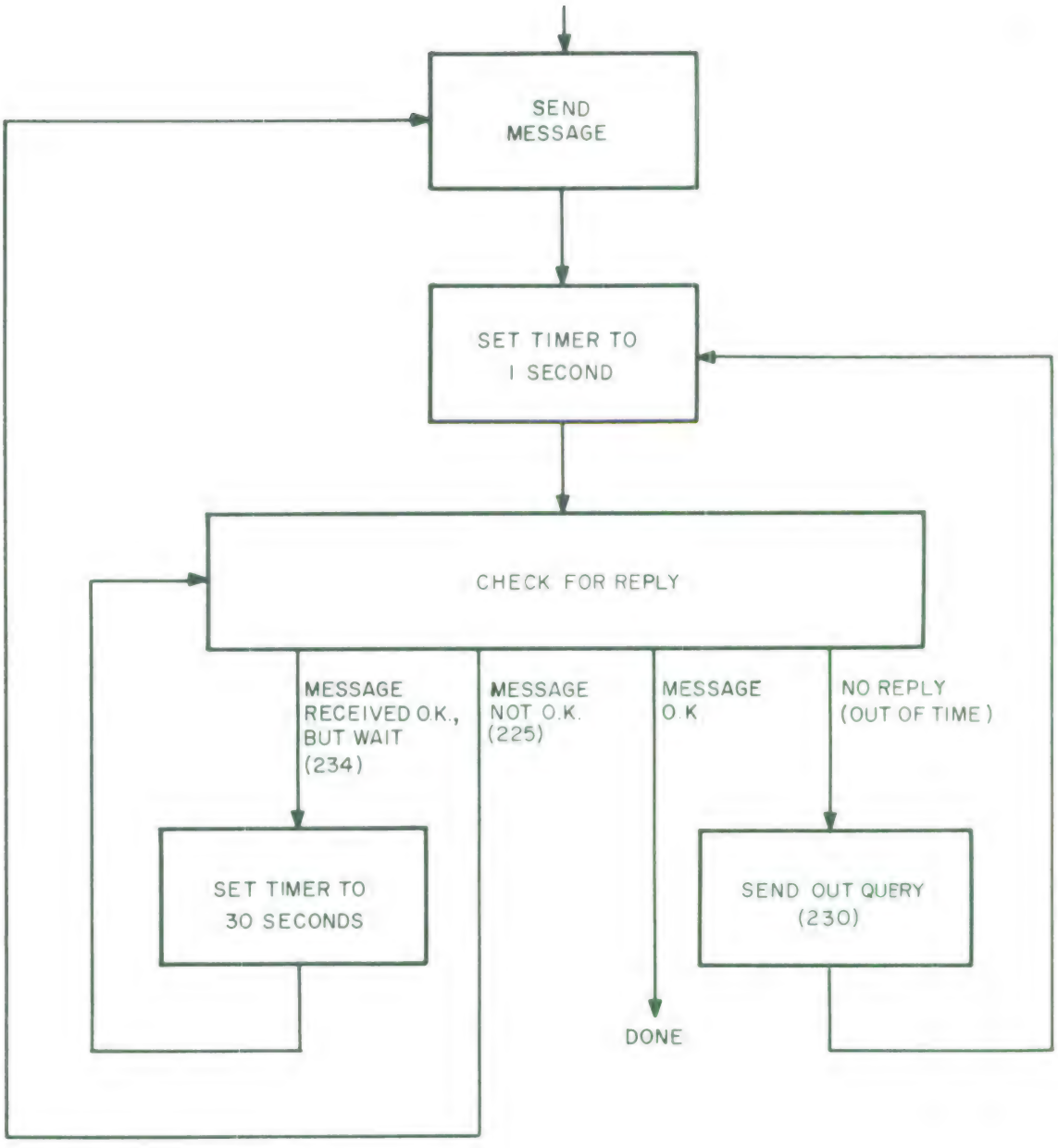

Figure 2. Transmission Message Protocol.

Chapter 4
Software

4.1 Monitor Considerations

SDC System

From the point of view of monitor programs, the simplest way of implementing a network is to handle the remote computers in the same manner as remote user stations. This is, in fact, the way the Q-32 handles the TX-2.

The SDC time-sharing system (see Section 2.3) runs on the Q-32 and uses the PDP-1 as a communications controller. No changes were made to the time-sharing monitor running on the Q-32.

In the PDP-1, programs were implemented for handling the protocol (see Section 3.5). When a message comes in (assuming it is accepted by the protocol as being correct), all characters except the text itself are stripped away, and the text is transmitted to the Q-32 as though it had come from a remote teletype.

Similarly, on output, the Q-32 ships characters to the PDP-1 as though for a remote teletype. The PDP-1, knowing that the characters are addressed at the TX-2, gives the characters to the message-protocol program.

For incoming calls, the above technique will automatically work with any time-sharing monitor. For outgoing calls, the technique will work provided the monitor allows a given user program to communicate with two user "consoles". One of these is the remote computer, and the other the local user

initiating the call. The only other requirement is that there be some mechanism for actually making the connection, if a dial-up system is being used. However, SDC has not debugged its automatic calling unit, hence no out-going calls have been initiated at SDC.

While the technique sketched above (which requires no monitor change) is easy to implement, it has, in the general case, serious drawbacks. For one thing, it does not allow for the communication of binary information, since a monitor will not, in general, allow a user program to transmit binary information to, or receive binary information from, a remote user station*. Thus, to provide for binary transmission, changes must be made to the monitor.

Secondly, the present technique allows the TX-2 to use only those Q-32 programs which receive all inputs from, and ship all output to, remote stations. An important class of programs which can therefore not be used is the class of display programs. Such programs do not ship their output as character strings to remote stations (in the conventional sense of keyboard-and-printer devices), but rather send special display files to display-generators**.

---

* Certain characters received from a user station are usually interpreted as special instructions to the monitor itself, and could occur by chance in a binary stream.

** In the case of the Q-32, display files are sent to display drums.

At the time the networking project started, no facilities existed at TX-2 for interfacing with common-carrier networks. These had to be implemented from scratch, and this opportunity was also used for making some further changes to the APEX time-sharing monitor for the purpose of handling the network in a somewhat more sophisticated manner than described above.

In APEX, the user program communicates with the monitor by making calls to the monitor (calls are similar to monitor commands). For purposes of networking, various new calls were implemented. These deal with dialing SDC, with establishing a connection to the 338, with shipping out a batch of alphanumeric or binary information, and with receiving a batch of alphanumeric or binary information from the link.

In addition, changes were made to APEX so that, under certain circumstances, some calls are reinterpreted by the monitor as having new significance. If a user program is being used by a remote computer, APEX takes cognizance of this fact in the following way: (a) If the program makes a "typeout" call to the monitor, the call is reinterpreted as a command to send out alphanumeric information over the link. (b) If this program makes a call that says "display this information on a display scope", the call is reinterpreted as a command to send out binary information over the link.

General Considerations

The design of network-and-time-sharing monitors must take the following considerations into account.

First of all, a message protocol must be implemented. This program is not part of the time-sharing monitor proper, but exists at the "periphery", or in a separate computer which does message-handling (the PDP-1 at the SDC system or the IBM 7750 at CTSS*).

Regarding the monitor itself, certain commands must be implemented for establishing the connection to the new computer and for allowing messages to be sent out and received by user programs. At a minimum, these commands should specify, aside from the address**, whether the message is alphanumeric or binary.

This much is obvious. What is less obvious is the fact that if programs are to be used on a given computer or by a remote computer with no change to these programs, then further sophistication must be added to the monitor. At a minimum (as done at SDC), the monitor must be able to route typeouts to the network message-protocol-handling routines for transmission to the remote computer, and vice-versa on typein. A more sophisticated monitor (as done on TX-2) should also automatically divert display information to the network when the display program is being called by a remote computer and be prepared to receive light-pen information. In the most general case, all commands given to the monitor by a user program should be examined when this program is used from a

---

* The Compatible Time-Sharing System at Project MAC, M.I.T.

** In the general case, the address includes the designation of (a) the computer, (b) whether monitor or user program, (c) if the latter, which particular user program, and (d) within a user program, which of several buffers. Further breakdown is required for monitor messages.

remote computer, and the command reinterpreted as necessary.

4.2 User Program Considerations

TX-2

We consider first the case in which TX-2 is the home computer, that is, the one initiating the network call. From the point of view of a user program running at TX-2, the network appears as follows.

The phone line is treated as a shared I/O device. The "device" may be used by only one user program at a time. The user program (via "calls", i.e., commands, to the APEX monitor) requests that communication be established with another computer. If successful, the phone line is reserved for that user program until the call is terminated. The content of transmitted messages is determined by the user program, but the monitor takes care of all message protocol, initiating retransmissions as necessary. Similarly, the user program can dispose of incoming messages in any way desired. When the user is through, his program relinquishes the phone line and it becomes available for another user.

If the connection is to SDC, it is the user program's responsibility to deal with the SDC monitor. That is, the user program must log itself in and cope with the usual responses of the SDC monitor.

If TX-2 is the remote computer, the APEX monitor will answer the phone and start up the login program. The user program that is called up from the remote computer will then automatically have all keyboard-typewriter I/O diverted to the network without requiring any action from the user program.

Similarly, the user program display calls will automatically be translated into messages for transmission to a remote computer. Not all of a user program's output, however, is diverted to the network. Any calls involving the printer or paper-tape reader will operate as usual (locally)*.

Thus, TX-2 user programs all of whose output is in the form of typeouts or displays and all of whose input is in the form of typeins may be used directly from a remote computer.

SDC

At SDC, user programs which are called from a remote computer have typeouts and typeins diverted to the network instead of a local terminal. No other special action is taken. Thus, SDC user programs whose I/O is in the form of typeouts and typeins may be used directly from a remote computer. The facility for having a user program call out has not yet been made operational.

It should be noted that the above describes the condition under which a program may be used directly. One always has the option, for example, of writing a special SDC user program (call it A) for the purpose of gaining remote access to an existing program (call it B), in case B does not type in and type out all of its I/O. In such a situation, A acts as an interface between the link and the desired program.

---

* The use of such local facilities may be denied to a remote user at the discretion of the monitor.

4.3 Existing User Programs and Demonstration Packages

TX-2

There did not exist, at TX-2, prior to the network, a program for allowing users, at their console, to type in algebraic expressions and obtain numerical answers for the values of the expressions. A very good set of arithmetic routines (Reckoner) did exist, so that the problem was reduced to that of writing a compiler from algebraic expressions into Reckoner code.

It was suggested that, while it would be a relatively difficult undertaking in machine code, such a translator could be written in LISP in a matter of a few hours. Since there is no LISP on the TX-2 but an excellent version runs on the Q-32, it was therefore decided to implement the translator through the network.

The program operates as follows (see printout, given in Fig. 3).

1. The user at a TX-2 console types CCA to log in.
2. He then types AT (algebraic translator). The AT program dials SDC, logs in, loads the LISP compiler. Since TX-2 cannot save any files on the Q-32, AT transmits a LISP source program to the Q-32. This source program is then compiled by the LISP compiler at the Q-32 into an object program.
3. AT now accepts input from the TX-2 typewritter. In the example of Fig. 3, the following is typed in:

```
X = 7.012345
(SIN X)*(SIN X) + (COS X)*(COS X).
```

The AT program thereupon types out the answer 1.0000000. To obtain the answer, AT transmitted the algebraic expression to the Q-32 for compilation into Reckoner code, received the code, executed this code on the TX-2 and typed out the answer. No significant delay between input of problem and output of answer was observed.

In the particular demonstration represented in Fig. 3, the set-up time, that is, the interval from log on at the TX-2 to the moment the system accepts algebraic strings from the console, was about five minutes. These five minutes include:

1. Selecting AT as the program to run
2. Dialing SDC
3. Logging in at SDC
4. Loading the LISP compiler at SDC
5. Transmitting the source program to be compiled at SDC
6. Compiling this source program

(All steps after the first are automatic.) It should be noted that the five minutes include the usual operating delays involved in using time-sharing systems (two of them in this case) in the middle of the day.

```
LOG IN CCA
          CCA IN ‖ 425 PGS ,  THU 16 FEB 67 1320.13
          BTR 2
AT
DIALING SDC
CONNECTED
1967 FEB 16 1021.9  LISP 1.5 M2.6 7JA13 RL1447
 (EX CD)
 TLST
 BOPS
 UOPS
 (COMP EVBT TEST)
```

```
X=7.012345

(SIN X)*(SIN X) +(COS X)*(COS X)
  1.0000000

X=1

Y=3

X/Y
  3.3333332×10^-1
```

Figure 3. Demonstration of Algebraic Translator

SDC

The facilities for using SDC as the home computer have not been debugged. Hence, no demonstrations initiated at SDC have been given.

338

To date, the 338 has been operated only as a remote TX-2 terminal; therefore, only one user program has been written for the DEC 338. This is a program which makes the 338 keyboard appear to be a TX-2 display*. Other user programs could, in principle, be written.

With the 338 program, most TX-2 user programs can be run from the 338. In particular, those programs which take their input from a Lincolnwriter and give their output to a Lincolnwriter or to a display device can be run; those programs which take input from a light-pen, however, cannot be run. The restriction on light-pen inputs stems from certain problems in the interaction of the networking software, the message protocol, and APEX. This restriction could be lifted at some future time, but some redesign would be required.

An interesting feature of 338 operation is the fact that, since the 338 has no bulk memory device, the 338 user program itself must be kept on TX-2. The start-up procedure is as follows.

First a bootstrap program is loaded into the 338 from local punched paper tape. The bootstrap program is then run, and causes an interchange of messages between the 338 and the TX-2 to take place; as a result, the 338 is logged in on the

---

* An analogous program has been written for the TX-2 which makes a TX-2 Lincolnwriter, by means of the network, appear to be an SDC teletype. To allow this program to be easily used, the 338 is equipped with a Lincolnwriter keyboard so that the special TX-2 character set is available to the user.

TX-2 under APEX. At this point, the full networking software is operational on the TX-2, but a minimal version (which computes no checksums, for example) is running on the 338. The bootstrap program then sends the command to the TX-2 to load and run a program called LOAD. The effect of this program is to ship to the 338 what may be called the "real loading program". This program now runs on the 338; its sole function is to accept certain commands from the 338 console indicating that certain files from the TX-2 should be shipped to the 338. (In practice it is always the same set of files, namely those comprising the single 338 user program plus the full 338 version of the networking software; in principle, as mentioned, there could be a variety.) The requested files are then shipped to the 338, replacing the loading program. These files contain the operating 338 programs.

Chapter 5
Operating Experience and Recommendations

5.1 Interface Hardware

The 338 uses standard interface hardware (see Section 2.4); the Q-32 uses only a slight modification of previously existing hardware (Section 2.3). Most of the experience in the design and fabrication of interface hardware has been gained on the TX-2.

Now that this TX-2 hardware is operational, there is some feeling that an alternative implementation might have been preferable: this alternative would have been to dedicate a small general-purpose computer to the task of interfacing with the communication lines. If such a separate buffering computer had been used, some of the advantages might have been the following:

(a) Shift registers would not have been required, since bit-by-bit operation would have been adequate to keep up with the data-rates under consideration.

(b) Double-ranked buffers would not have been necessary.

(c) Control logic would not have been needed; in effect, the program on the buffering computer would have replaced the logic.

(d) The system might have been operational sooner (although there is no guarantee of this).

It may also be pointed out that if such buffering computers were used in this fashion at every computer installation in

the network, certain additional advantages would accrue*. These are that a unified implementation could be specified for the communication portion of the network, so that this portion would be independent of the particular characteristics of the individual computers. Changes in protocol would be easier to accomplish, since the same protocol-handling software would be used throughout. A decreased load on each of the main computers would also result.

The principal disadvantage of this approach is increased cost.

---

* This point was first made by Wesley Clark of Washington University and represents the view adopted by the Advanced Research Projects Agency in the creation of its next-generation computer network [10].

5.2 Communications

A series of tests were made on the TX-2 - Q-32 system. Data on the number of calls placed, the number successfully connected, and the average time to achieve a successful connection were measured in four successive two-week periods. The results are as follows.

	Period				Total
	1	2	3	4	
No. of calls in sample	34	33	6	21	94
No. of calls successfully connected	25	28	6	19	78
% successfully connected	73%	85%	100%	90%	83%
average time for successful connection (seconds)	19.5	19.5	19.5	21.1	19.9

After some initial problems at the time of installation, the system was felt to operate with adequate reliability. Period 1 in the above table coincided with a period of intermittent trouble with equipment at SDC. This fact may account for the lower proportion of successfully connected calls in Period 1.

The average time to achieve a connection is relatively long (about 20 seconds) but, since the calls themselves tend to be very much longer (see below), this time is not unacceptable. (It may be that the emphasis on long calls is a result of the long time needed to achieve a connection.)

If networking operation were to change in such a way that very

short calls became common*, then a way to speed up the process of making the connection would have to be found.

Tests were made on the TX-2 - 338 system at 1200 bits per second using the asynchronous modems. The results are given below. (These figures were gathered during actual use of the network.)

Number of calls in sample: 29
Average duration of call: 1355 seconds (22 minutes, 35 seconds)
Average number of data characters transmitted per call: 5564
Average number of messages transmitted per call: 158
Average number of bad messages** transmitted per call: 1
Character-error rate***: 0.014%
Efficiency****: 3%

These figures show that the error rate on the whole is quite satisfactory: in terms of characters, one finds about one error in 7000. The message protocol has no trouble in detecting and correcting these errors. The highest character error rate observed in any one call in the sample of 29 calls was 0.09%.

Additional tests were conducted on the TX-2 - Q-32 system at 1200 bits per second using the asynchronous modems. The results were as follows.

---

* As would be the case in a I'll-hang-up-and-you-call-me-back-when-you-get-the-answer mode of operation.

** A bad message is one to which a NACK reply is received. (See Section 3.5)

*** Assumes one bad character per bad message.

**** Efficiency is defined as the ratio of the number of characters transmitted to the number of characters that could have been transmitted under continuous transmission.

Number of calls in sample: 17
Average duration of call: 1494 seconds (24 minutes, 54 seconds)
Average number of data characters transmitted per call: 4800
Average number of messages transmitted per call: 111
Average number of bad messages transmitted per call: 12
Character-error rate: 0.25%
Efficiency: 3%

These figures are similar to the preceding ones, with one glaring exception: the error rate is much higher. To check into this further, "loop tests" were run. In a loop test, messages are sent out by TX-2 to the Q-32, but are mirrored back to the East coast at the Q-32 modem without ever entering the Q-32. Under these conditions, the observed error rate was very much lower, being similar to the error rate observed when communicating with the 338*. One might surmise that a program error at SDC accounts for the high error rate.

Efficiency of utilization of the line is seen to be extremely low. Low efficiency will generally be observed when a person at a terminal is directly involved (as was the case during the tests). Thus, the 338 was used only as a remote terminal of the TX-2; the Q-32 was used to service a person at a TX-2 terminal. The low efficiency therefore reflects the person's slow reaction time.

While the overall efficiency is low, the peak rate is occasionally quite high. Thus, when the 338 initiates a start-up procedure (see Section 4.3), substantial files are transmitted and the transmission rate is high.

---

* Out of 5632 messages transmitted over several trials, two errors were observed.

High overall efficiency of the line could be achieved by a multiplexing scheme, in which messages pertaining to different user programs make use of the same line. Any such scheme, however, would have to be designed to cope with the high peak rates that can occur.

Ultimately, one might feel that the common carrier should charge on the basis of the amount of information transmitted rather than on the basis of the amount of time connected. Such a scheme would, in effect, push the problem of multiplexing efficiency back in the lap of the carrier where, it might be argued, it really belongs.

5.3 Communication Protocol

A. As the message format and protocol is currently organized, there is a single header character (see Section 3.4); in the case of binary messages, this character is an octal 221 or an octal 232 according as the message is addressed to the monitor program at the receiver or to a user program (201 or 202 in the case of alphanumeric messages). This distinction is insufficient; the message format and protocol should be redesigned so as to include a large variety of destinations. In general, if a time-sharing monitor on a given computer allows a user program to accept inputs from n different devices (light-pen, mag-tape, paper tape, cards, etc.), then a remote computer communicating with this program should have the facility to address messages to n individual buffers. Furthermore, it may be desirable (as is the case in communicating with SDC) to be able to address messages to the monitor. In general, therefore, the message format should allow messages to be addressed to any one of a large number of destinations, each of which is individually buffered, and which is assigned a meaning by the receiving computer. In the case of an expanded message destination capability, the protocol must also allow the calling computer to be informed in the event that the destination code is in error.

In the present format there is only one message (octal 234) which says "message received OK but buffers are full--wait before sending next message". This should be enlarged to encompass a variety of alternatives so that messages of the following form could be sent: "message received OK but buffers of type k are full--wait before sending next message to buffers of type k".

B. There is a further drawback to the present protocol which should be mentioned. In the present way of communicating, the transmitter must wait to receive an acknowledgment before transmitting the next message. This technique was adopted for maximum simplicity, particularly in view of the small amount of space available at SDC for the network software. However, the present procedure tends to be quite slow--much slower than would be necessary in future implementations. It would be entirely possible to design a message format and protocol which would not require the transmitter to wait for an acknowledgment before sending the next message*. In order to accomplish a transmission in a continuous mode, it will be necessary to assign numbers to messages; the receiving computer can then acknowledge messages by their number. If a negative acknowledgment for a message sent several messages ago is received, this particular message will be retransmitted.

c. As seen in Chapter 3 the format and protocol used in the experimental network is not ASCII conforming, in that it uses a count character to indicate the number of characters in the message, thereby being able to transmit full binary information at 8 bits per character. This technique has worked out well, and we recommend it highly.

---

* Indeed such a format and protocol was designed for the present network, but was rejected as too complex for the initial attempt.

References

[1] T. Marill, "A Cooperative Network of Time-Sharing Computers: Preliminary Study", Technical Report No. 11, Computer Corporation of America, Cambridge, Massachusetts (1966).

[2] T. Marill and L.G. Roberts, "Toward a Cooperative Network of Time-Shared Computers", Proc. AFIPS Fall Joint Computer Conference, pp. 425-431 (1966).

[3] J. Heller, Catch-22, N.Y.: Dell Publishing Company, 1955.

[4] "Proposed Revised American Standard Code for Information Interchange", Communications of the Association for Computing Machinery, Vol. 8, No. 4, April 1965, pp. 207-214.

[5] "Character Structure and Character Parity Sense for Serial-by-Bit Data Communication in the American Standard Code for Information Interchange", Communications of the Association for Computing Machinery, Vol. 8, No. 9, September 1965, p. 553.

[6] "Control Procedures for Data Communication--An ASA Progress Report", Communications of the Association for Computing Machinery, Vol. 9, No. 2, February 1966, pp. 100-107.

[7] Communications of the Association for Computing Machinery, Vol. 8, No. 4, April 1965.

[8] A.K. Bhushan and R.H. Stotz, "Message Format and Protocol for Inter-Computer Communication", Electronic Systems Laboratory, M.I.T., Cambridge, Massachusetts, Project MAC Memorandum MAC-M-351, June 16, 1967.

[9] A.A. Alexander, R.M. Gryb, D.W. Nast, "Capabilities of the Telephone Network for Data Transmission", The Bell System Technical Journal, Vol. 39, pp. 431-476, May 1960 (see Fig. 37).

[10] "ARPA Seeks Small Machines for Time-Shared EDP Net", Electronic News, May 27, 1968, p. 38.

[11] J.W. Forgie, "A Time-and-Memory-Sharing Executive Program for Quick-response On-line Applications", Proc. AFIPS Fall Joint Computer Conf., Vol. 27, part 1, pp. 599-609, 1965. Spartan Books.

[12] J.L. Eisenbies, "Conventions for Digital Data Communication Link Design", IBM Systems Journal, Vol. 6, No. 4, pp. 267-302, 1967.

DOCUMENT CONTROL DATA – R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Computer Corporation of America Under Purchase Order C-538 to Lincoln Laboratory, M.I.T.

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP
None

3. REPORT TITLE
An Experimental Computer Network

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Final Report covering 1 January 1967 through 31 March 1969

5. AUTHOR(S) *(Last name, first name, initial)*
None given

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
March 30, 1969	57	12

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
AF 19 (628)-5167 b. PROJECT NO. ARPA Order 691 c. Purchase Order C-538 d.	None 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* ESD-TR-69-74

10. AVAILABILITY/LIMITATION NOTICES
This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Advanced Research Projects Agency, Department of Defense

13. ABSTRACT

The concept of a computer network is reviewed. Within such a network, a user at any one installation has immediate access to the hardware, software, and data at all other installations. The various installations need not be program-compatible.

The implementation of a small experimental network based on these concepts is discussed. The details of the implementation and the experience gained in the operation of this experimental network are described.

14. KEY WORDS
computer computer network